МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інженерії програмного забезпечення, канд. техн. наук, доцент
_____ Є. О. Давиденко
«___»_____ 2023р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Мобільна гра в жанрі Action на рушії Unity

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ – 409.21910904

Студент _____К. Б. Валітов
«___»_____ 2023р.

Керівник PhD, ст. викладач _____ І. О. Кандиба
«___»_____ 2023р.

Консультант канд. техн. наук, доцент _____А. О. Алексєєва
«___»_____ 2023р.

Миколаїв – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені
Петра МогилиФакультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри інженерії
програмного забезпечення, доцент, канд.
тех. наук.
_____Є. О. Давиденко
«_____»____20__р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи бакалавра

Видано студенту групи 409 факультету комп'ютерних наук

_____Валітову Кирилу Борисовичу_____

(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи
_____«Мобільна гра в жанрі Action на рушії Unity»_____

Затверджена наказом по ЧНУ від «17» _____березня____2023р. №__60__

2. Строк представлення кваліфікаційної роботи «___»___2023р.

3. Очікуваний результат роботи та початкові дані якщо такі потрібні
____розробка та реалізація мобільної гри в жанрі Action_____

4. Перелік питань, що підлягають розробці:

Ознайомлення з теорією та аналіз теми, розробка концепту гри, розробка технічної частини гри, тестування та відлагодження гри, підготовка звіту та захисту.

5. Перелік графічних матеріалів

Презентація

6. Завдання до спеціальної частини

Аналіз охорони праці на робочих місцях фахівців з інформаційних технологій

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Алексєєва А.О.	екології	Охорона праці і здоров'я

Керівник роботи PhD, ст викладач Кандиба Ігор Олександрович _____

(підпис)

Завдання прийнято до виконання Валітов Кирило Борисович _____

(підпис)

Дата видачі завдання «_»____20___р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: «Мобільна гра в жанрі Action на рушії Unity»

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРБ	10.03.2023р.	17.03.2023р.	виконано
2.	Ознайомлення з літературою та аналіз теми	18.03.2023р.	01.04.2023р.	виконано
3.	Складання календарного плану КРБ	02.04.2023р.	03.04.2023р.	виконано
4.	Аналіз предметної області	04.04.2023р.	11.04.2023р.	виконано
5.	Розробка геймплею та рівнів	12.01.2023р.	19.04.2023р.	виконано
6.	Розробка персонажів та ефектів	20.04.2023р	27.04.2023р	виконано
7.	Реалізація геймплею та рівнів	28.04.2023р.	05.05.2023р.	виконано
8.	Реалізація персонажів та ефектів	06.05.2023р.	13.05.2023р.	виконано
9.	Перевірка функціональності гри	13.05.2023р.	15.05.2023р.	виконано
10.	Оцінка ефективності гри на різних мобільних пристроях	15.05.2023р.	17.05.2023р.	виконано
11.	Розробка спеціальної частини з охорони праці	18.05.2023р.	21.05.2023р.	виконано
12.	Відгук керівника КРБ	22.05.2023р.	23.05.2023р.	виконано
13.	Оформлення КРБ та презентації	24.05.2023р.	27.05.2023р.	виконано
13.	Попередній захист	28.05.2023р.	29.05.2023р.	виконано
14.	Рецензування	30.05.2023р.	01.06.2023р.	виконано
15.	Завершення оформлення КРБ та презентації	02.06.2023	19.06.2023	виконано
16.	Захист кваліфікаційної роботи	26.06.2023	27.06.2023	виконано

Розробив студент Валітов К. Б. _____

(прізвище, ім'я, по батькові) *(підпис)*

«__» _____ 20__ р.

Керівник роботи PhD, ст. викладач Кандиба І. О. _____

(посада, прізвище, ім'я, по батькові) *(підпис)*

«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Мобільна гра в жанрі Action на рушії Unity»

Студент 409 гр.: Валітов Кирило Борисович

Керівник: PhD, ст. викладач Кандиба Ігор Олександрович

Розробка якісного інтерактивного продукту на рушії Unity надзвичайно важливою для бізнесу, що працює в галузі розробки ігор. Розуміння особливостей цього жанру та використання можливостей рушія Unity для реалізації інноваційних ідей є ключовим фактором успіху на ринку мобільних ігор.

Мета: є популяризація ігрового жанру Action на мобільних телефонах, шляхом розробки мобільної гри на рушії Unity.

Об'єктом кваліфікаційної роботи є процес розробки ігор на рушії Unity.

Предметом кваліфікаційної роботи є інструментарій розробки гри в жанрі Action на рушії Unity.

Для досягнення цієї мети необхідно вирішити наступні завдання:

Дослідити особливостей жанру Action та провести аналіз сучасних мобільних ігор цього жанру;

Проаналізувати можливості рушія Unity для розробки мобільних ігор та використання його функцій для реалізації особливостей гри в жанрі Action;

Спроєкутвати гру в жанрі Action;

Розробка дизайну гри, включаючи геймплей, рівні, персонажів та ефекти;

Реалізація гри на рушії Unity та тестування її функцій та ефективності.

Пояснювальна записка кваліфікаційної роботи бакалавру складається з вступу, трьох розділів та додатків.

У вступі визначається актуальність теми, що приймається за мету та невеликий огляд поставленої задачі, предмет дослідження та об’єкт дослідження.

У першому розділі описується аналітична частина, тобто огляд існуючих аналогів. Визначається основна особливість ігор, завдяки чому було сформовано загальне розуміння предметної області.

У другому розділі описується процес розробки та вибір мови програмування, середовище розробки, ігровий двигун. Розробка UML-діаграм та опис інтерфейсів.

У третьому розділі демонструється проведена робота з кодування та тестування, крім того описується розробка ігрового меню та налаштувань для зручного використання.

У висновках проводиться аналіз роботи та отриманих результатів.

Кваліфікаційна робота бакалавра викладена на 51 сторінку, вона містить 4 розділи, 19 ілюстрацій, 7 таблиці, 16 джерел в переліку посилань.

Ключові слова: Unity, action, мобільна гра.

ABSTRACT

of the Bachelor’s Thesis

"Mobile Action Game on Unity Engine"

Student of group 409: Valitov Kyrylo Borysovych

Supervisor: PhD, Senior Lecturer Kandyba I. O.

Developing a quality interactive product powered by Unity is extremely important for a game development business. Understanding the specifics of this genre and using the capabilities of the Unity engine to implement innovative ideas is a key success factor in the mobile game market.

The goal is to popularize the Action game genre on mobile phones by developing a mobile game on the Unity engine.

The object of the qualification work is the process of game development on the Unity engine.

The subject of the qualification work is the toolkit for game development in the Action genre on the Unity engine.

To achieve this goal, the following tasks must be solved:

Investigate the features of the Action genre and conduct an analysis of modern mobile games of this genre;

Analyze the capabilities of the Unity engine for developing mobile games and use its functions to implement features of the game in the Action genre;

Develop a game in the Action genre;

Game design development, including gameplay, levels, characters and effects;

Implementation of the game on the Unity engine and testing of its functions and effectiveness.

The explanatory note of the bachelor's thesis consists of an introduction, three sections and appendices.

The introduction determines the relevance of the topic, which is taken as the goal and a brief overview of the task, the subject of research and the object of research.

The first section describes the analytical part, that is, a review of existing analogues. The main feature of games is determined, thanks to which a general understanding of the subject area was formed.

The second chapter describes the development process and choice of programming language, development environment, game engine. Development of UML diagrams and description of interfaces.

The third chapter shows the coding and testing work done, and also describes the development of the game menu and settings for easy use.

In the conclusions, an analysis of the work and the obtained results is carried out.

The bachelor's qualification work is laid out on 58 page, it contains 4 sections, 19 illustrations, 7 tables, 16 sources in the list of references.

Keywords: Unity, action, mobile game.

ЗМІСТ

ВСТУП

На сьогоднішній день ринок мобільних ігор швидко росте, і жанр Action є одним з найбільш популярних. Це робить розробку якісного інтерактивного продукту на рушії Unity надзвичайно важливою для бізнесу, що працює в галузі розробки ігор. Розуміння особливостей цього жанру та використання можливостей рушія Unity для реалізації інноваційних ідей є ключовим фактором успіху на ринку мобільних ігор. Таким чином, розробка мобільної гри в жанрі Action на рушії Unity є не лише актуальною, але й може бути комерційно вигідною.

Об'єктом кваліфікаційної роботи є процес розробки ігор на рушії Unity.

Предметом кваліфікаційної роботи є інструментарій розробки гри в жанрі Action на рушії Unity.

Метою кваліфікаційної роботи є популяризація ігрового жанру Action на мобільних телефонах, шляхом розробки мобільної гри на рушії Unity.

Для досягнення цієї мети необхідно вирішити наступні завдання:

1. Дослідити особливостей жанру Action та провести аналіз сучасних мобільних ігор цього жанру.
2. Проаналізувати можливості рушія Unity для розробки мобільних ігор та використання його функцій для реалізації особливостей гри в жанрі Action.
3. Спроєкутвати гру в жанрі Action.
4. Розробка дизайну гри, включаючи геймплей, рівні, персонажів та ефекти.
5. Реалізація гри на рушії Unity та тестування її функцій та ефективності.

Метою спеціального розділу про охорону праці є дослідження норм охорони праці, які пов’язані з роботою розробника програмного забезпечення.

1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ

1.1 Жанри мобільних ігор

Мобільні ігри на сьогоднішній день відіграють важливу роль як в економічному, так і в культурному аспектах. Вони стали невід'ємною частиною індустрії розваг та креативного сектора економіки. Однак, естетика і технології, що використовуються у мобільних іграх, знаходять застосування в значно ширшому спектрі сфер: вони допомагають у процесі навчання дітей, молоді та дорослих як у школі, так і на робочому місці, надаючи веселий спосіб освоєння складного матеріалу. Крім того, мобільні ігри можуть бути використані для мотивації пацієнтів у медичних та терапевтичних установах, створюючи стимулююче середовище для досягнення позитивних результатів. Таким чином, мобільні ігри виходять за межі простого розважального середовища і знаходять широке застосування у різних сферах життя.

У світі мобільних ігор існує безліч різноманітних жанрів, які втілюють в собі різноманітні стилі, геймплей та ігрові механіки. Від захоплюючих рольових пригод і епічних стратегій до швидкої аркади і складної головоломки - мобільні ігри пропонують безмежну кількість можливостей для розваги і відпочинку.

Залежно від свого настрою, користувач може відчути адреналін в гонках, потренувати свою логіку в головоломках або побудувати свій власний світ у стратегічних іграх.

Для прикладу розглянемо декілька основних жанрів[1-5]:

- рольові ігри;
- головоломки і логічні ігри;
- аркадні ігри;
- гонки і спорт;
- стратегії;
- action.

Рольові ігри (RPG) – це жанр, який переносить гравців у захоплюючий світ фантазії і пригод. У цих іграх користувач може відчути себе в ролі героя, який веде епічну кампанію, розвивається і виграє битви з різними супротивниками. RPG надають гравцям можливість вибирати свого персонажа, його характеристики, вигляд і навички.

У світі RPG кожне рішення гравця має значення. Ви можете взаємодіяти з іншими персонажами, приймати рішення, які впливають на сюжет та розвиток історії. Захоплюючі квести, завдання і діалоги розкриваються по ходу гри, дозволяючи гравцю погрузитися в унікальну іммерсивну атмосферу ігрового світу.

У RPG важливо розвивати свого персонажа. За допомогою набуття досвіду та виконання завдань, гравець отримує можливість покращити навички, отримати нове спорядження та відкривати більше можливостей. Гравець може вибрати роль воїна, мага, лучника або багатьох інших і підвищити їх вміння, щоб стати непереможними героями.

Приклад мобільних ігор у жанрі RPG: The Elder Scrolls: Blades, Final Fantasy XV Pocket Edition, Summoners War, Star Wars: Knights of the Old Republic.

Жанр головоломок і логічних ігор на мобільних пристроях - це світ, де гравці стають справжніми майстрами розумових викликів. Ці ігри перенесуть гравця в захоплюючі лабіринти складних загадок, головоломок і завдань, які вимагають логічного мислення, кмітливості та творчого підходу.

В цьому жанрі існує широкий спектр ігор, починаючи від класичних головоломок, таких як "тетріс" і "судоку", до більш складних ігор з унікальними геймплейними механіками. Гравець має розгадувати логічні загадки, збирати пазли, шукати сховані об'єкти або вирішувати криптографічні головоломки, щоб просуватися далі в грі.

Ці ігри не лише веселі, але й корисні, оскільки вони розвивають креативність гравця, аналітичні здібності, логічне мислення та стратегічне

планування. Вони стимулюють гравця вдумливо аналізувати ситуацію, шукати рішення та розвивати нові підходи до проблем.

Приклад мобільних ігор у жанрі головоломки і логічні ігри: The Room, Monument Valley, The Witness, Cut the Rope, Lara Croft GO.

Аркадні ігри – це жанр мобільних ігор, що включають в себе швидкі ігрові сесії з простими правилами та ігровим процесом. Цей жанр ігор почав свій шлях від ігрових залів, де гравці мали можливість відчути адреналін із змагань за рекордні результати. Сьогодні аркадні ігри є дуже популярними серед мобільних гравців, і вони мають широку аудиторію.

Характерними ознаками аркадних ігор є швидкість геймплею, прості правила та елементи, які можуть змінюватися для кожної ігрової сесії. Ці ігри можуть бути різноманітні – від класичних арканоїдів та тетрісів до сучасних раннерів та файтингів.

Приклад мобільних ігор у жанрі аркади: Angry Birds, Fruit Ninja, Candy Crush, Flappy Bird, Pac-Man.

Жанр гонок і спорту в мобільних іграх завжди викликає піднесення та адреналін. Він пропонує гравцям можливість відчути швидкість, напруженість змагань і радість перемоги, все це на долоні вашої руки.

У цьому захоплюючому жанрі гравець зможе випробувати себе в ролі гонщика на найшвидших трасах, керуючи автомобілями, мотоциклами або навіть велосипедами. Захоплюючі графіка та реалістична фізика дозволяють гравцю відчути кожну криву, кожен скрутний поворот і швидкісний розгін.

Аркадні гонки пропонують гравцям шалений швидкісний досвід з використанням різноманітних механік, щоб поборотися за перші місця та побити рекорди. Гравець зможе випробувати свою реакцію та швидкість на трасах, які переповнені різноманітними перешкодами.

Окрім того, жанр гонок і спорту включає ігри, які симулюють різні види спорту, такі як футбол, бейсбол, баскетбол і навіть серфінг. Ви зможете стати

членом команди, взяти участь у змаганнях і використовувати свої навички для досягнення перемоги.

Приклад мобільних ігор у жанрі гонки і спорт: Asphalt 9: Legends, FIFA Mobile Football, NBA 2K Mobile Basketball.

Жанр стратегій в мобільних іграх відкриває перед гравцями широкий світ можливостей та вимагає стратегічного мислення, планування і управління ресурсами. В цьому жанрі гравець може стати володарем власної імперії, будувати та розвивати міста, керувати армією або керувати економікою цілого народу.

У світі стратегій гравець має обрати різні сценарії – від середньовічних воєн і фантастичних світів до сучасних політичних конфліктів. Кожна гра в цьому жанрі пропонує свій унікальний підхід та геймплей. Ви можете відчути себе в ролі великого полководця, який веде свою армію до перемоги, або відправитися у подорож в необережний світ політичних інтриг і дипломатії.

Приклад мобільних ігор у жанрі стратегія: Clash of Clans, Civilization VI, Plague Inc, XCOM: Enemy Within.

Жанр Action (екшн) в мобільних іграх пропонує шалений ритм, захоплюючі пригоди та безліч екшну. Це жанр, що ставлять перед гравцем виклик у вигляді швидких реакцій, неймовірних відчутті та боротьби зі світом, повним небезпек і ворогів.

У світі ігор Action гравець може стати справжнім героєм, який вирушає на місію порятунку, протистояти злим силам або відправитися на війну. Вам належить битися з численними ворогами, використовуючи різноманітну зброю, від ножів і пістолетів до магічних заклять і суперсили. Гравець має пройти безліч рівнів, виконувати завдання і виявити свою вправність у битві.

У мобільних іграх Action гравець зможе зануритися в світ небезпечних пригод, використовуючи інтуїцію, стратегічне мислення і швидкі рефлекси. Гравець зможе відчути себе справжнім героєм, який протистоїть всім випробуванням і небезпекам, що трапляються на його шляху.

Приклад мобільних ігор у жанрі Action: Shadow Fight 3, PUBG Mobile, Call of Duty Mobile, Injustice 2.

1.2 Аналіз аналогів

Alien Shooter

Таблиця 1.1 – Характеристика гри Alien Shooter

Назва характеристики	Опис
Назва	Alien Shooter
Розробник	Sigma Team
Архітектура	Гра є двомірною грою з видом зверху.
Мова реалізації	C++ з використанням DirectX для графічної обробки.
Перелік функцій, характеристик	– Гравець керує персонажем, який переміщується в просторі, стріляє ворогів та збирає бонуси. – Гра містить більше 50 видів зброї, включаючи важку артилерію та кібернетичний екзоскелет. – Гра містить елементи рольової гри, такі як можливість прокачувати характеристики персонажа, купувати нову зброю та екіпірування. – Гра має 10 рівнів складності, що різняться кількістю ворогів та їхньою силою. – Гра містить елементи рольової гри, такі як можливість прокачувати характеристики персонажа, купувати нову зброю та екіпірування. – Гра має мультиплеєрний режим, де гравці можуть битися проти ворогів разом.

Кінець таблиці 1.1

Аналіз переваг та недоліків	Переваги: – Ігровий процес відбувається в режимі реального часу, що забезпечує високу динаміку гри. – Великий асортимент зброї та бонусів робить гру цікавою та різноманітною. – Можливість прокачувати характеристики та купувати нове обладнання дає гравцям більшу вільність у виборі стратегії гри. – Гра має мультиплеєрний режим, що дозволяє гравцям грати разом. Недоліки: – Гра має відносно просту графіку та малу кількість деталей. – Деякі гравці можуть вважати, що гра стає одноманітною після деякого часу.

Рисунок 1.1 – Інтерфейс гри Alien Shooter

Xenowerk

Таблиця 1.2 – Характеристика гри Xenowerk

Назва характеристики	Опис
Назва	Xenowerk
Розробник	Pixelbite
Архітектура	Unity
Мова реалізації	C#
Перелік функцій, характеристик	– Боротьба зі зловісними мутантами: Гравці візьмуть на себе роль науковця, що проникає в лабораторію, заражену небезпечними мутантами. Вони повинні битися з цими ворогами та досліджувати лабіринти лабораторії. – Різноманітна зброя і спеціальні уміння: Гравці мають доступ до широкого вибору зброї та спеціальних умінь, які допомагатимуть їм в боротьбі з мутантами. Вони можуть покращувати свою зброю та розвивати свої уміння протягом гри. – Система досягнень, які стимулюють гравців до досягнення певних цілей.
	– Рівні та досягнення: Гра має декілька рівнів складності, які гравці можуть пройти. Крім того, є система досягнень, які стимулюють гравців до досягнення певних цілей. – Графіка та звук: Xenowerk має вражаючу графіку та атмосферний звуковий супровід, які допомагають зануритися в науково-фантастичний світ гри.

Кінець таблиці 1.2

Аналіз переваг та недоліків	Переваги: – Висока якість графіки і звукового супроводу, що допомагає створити настрій гри. – Різноманітність зброї та умінь, що додають глибини геймплею і стратегічного елементу. – Інтенсивні битви з мутантами, що забезпеч ують екшен і адреналін під час гри. – Цікавий сюжет та атмосфера науково-фантастичного світу, які захоплюють гравця і тримають його у напрузі. Недоліки: – Можлива повторюваність геймплею після тривалої гри, що може зменшити зацікавленість. – Обмежена кількість різних локацій та ворогів, що може призвести до відчуття монотонності. – Можливість нестабільної роботи гри на деяких мобільних пристроях з менш потужними характеристиками.

Рисунок 1.2 – Інтерфейс гри Xenowerk

Kill Deal

Таблиця 1.3 – Характеристика гри Kill Deal

Назва характеристики	Опис
Назва	Kill Deal
Розробник	Frosty Elk AB
Архітектура	Unity
Мова реалізації	C#
Перелік функцій, характеристик	– Гравці можуть вибирати з різних видів зброї, включаючи пістолети, гранати та автоматичні гвинтівки. – Гра пропонує кілька рівнів складності, що дозволяє гравцям з різним досвідом гри знайти відповідний виклик. – Гра пропонує онлайн-кооперативний режим, що дозволяє гравцям грати з друзями у режимі реального часу. – Гра має систему прокачки персонажа, яка дозволяє гравцям покращувати свої навички та зброю під час гри. – Гра має різноманітні місії та завдання, що забезпечують довгий та різноманітний геймплей.
Аналіз переваг та недоліків	Переваги: – Інтенсивна та захоплююча гра з відкритим світом та різноманітними завданнями. – Різноманітний вибір зброї та рівнів складності, що дозволяє гравцям з різним досвідом знайти свій рівень виклику.

Кінець таблиці 1.3

	– Онлайн-кооперативний режим, що дозволяє гравцям грати з друзями та спільно розвивати свої навички. Недоліки: – Гра може мати деякі технічні проблеми, такі як зависання або проблеми зі з'єднанням. – Гра може виявитися малоцікавою для тих, хто не цікавиться шутерами або постапокаліптичними іграми.

Рисунок 1.3 – Інтерфейс гри Kill Deal

1.3 Специфікація вимог до мобільної гри в жанрі Action

Призначення системи: є створення ігрового середовища, в якому гравці зможуть брати участь у захоплюючих боях та існувати, відчуваючи повну іммерсію та отримуючи задоволення від гри.

Сфера застосування: сфера розваг, реклама та маркетинг, розвиток ігрової індустрії.

Характеристики користувачів: користувачі віком від 16 років, повинні мати смартфон та доступ до мережі Інтернет.

Функції системи:

1. Система меню повинна містити список доступних режимів гри.
2. Система руху гравця.
3. Система зброї.
4. Система збору та пошуку ресурсів.
5. Система звуку та музики.
6. Система бойвих механік.
7. Система використання ресурсів.
8. Система інвентарю.
9. Система удосконалення персонажу та магазину.
10. Система збереження гри.

Вимоги до технічного забезпечення:

1. Операційна система: Android.
2. Процесор: кількість ядер – 4, частота – 2.0 ГГц + 1.8 ГГц.
3. Вільне місце на девайсі: 5ГБ.
4. Оперативна пам’ять: 4ГБ.

Архітектура програмної системи: складається з клієнтської частини.

Системне програмне забезпечення:

Для розробки ігрового застосунку на Unity необхідно встановити такі мінімальні системні програмні засоби, як:

1. Операційна система: Windows 7 SP1+, macOS 10.12+, або Ubuntu 16.04+.
2. Процесор: SSE2-сумісний процесор з тактовою частотою 1,8 ГГц або вище.
3. Оперативна пам'ять: мінімум 4 ГБ RAM.
4. Графічна карта: сумісна з DirectX 11 або OpenGL 3.2.
5. Місце на диску: мінімум 10 ГБ вільного місця.
6. Версія Unity: встановлення останньої версії Unity Hub та вибір відповідної версії Unity для розробки гри.

Мова і технологія розробки ПЗ: Рекомендовано використовувати мову програмування C#.

Інтерфейс користувача: Інтерфейс користувача має бути інтуїтивно зрозумілим та зручним для використання. Необхідно створити добре продуманий інтерфейс, який дозволить користувачам легко керувати персонажем, взаємодіяти з ігровим світом та керувати настройками гри.

Висновки до розділу 1

У розділі 1 було проведено аналіз предметної сфери розробки ігрових застосунків. З’ясовано значення поняття «аркада», визначення наведеного терміну допомогло з’ясувати його значення та актуальність.

Обрано три аналоги гри-застосунку, що були проаналізовані окремо та визначено їх характерні риси, такі як розробник та видавник, мова випуску, перелік функцій та інше. Виокремлено характерні переваги та недоліки ігрових застосунків, що були розглянуті.

У заключній частині розділу 1 сформульовано специфікацію вимог для мобільної гри в жанрі Action на рушії Unity.

2 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ МОБІЛЬНОЇ ГРИ

2.1 Створення діаграм прецедентів

Діаграма варіантів використання відображає взаємодію між акторами (користувачами або зовнішніми системами) та самою системою. Вона показує, як користувач взаємодіє з системою, які дії він виконує, та як система відповідає на їх запити [9].

Діаграма варіантів використання представлена на рисунку 2.1.

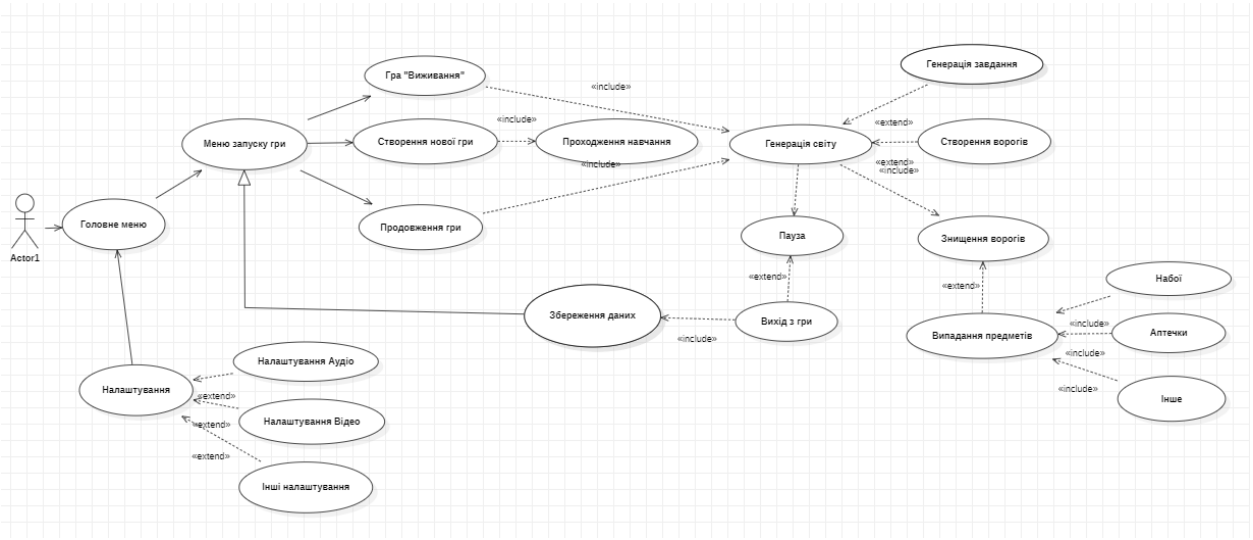

Рисунок 2.1 – Діаграма варіантів використання

Гра починається з головного меню [6-8], з якого можна перейти в налаштування, або у меню створення гри. В меню створення гри, якщо користувач вже грав раніше, тоді в нього є збережені дані минулих дій і він має змогу продовжити грати далі, в інакшому випадку йому необхідно створити нову історію персонажу.

Після запуску гри та генерації світу, користувачу буде надане завдання яке він має завершити щоб пройти початковий рівень. В продовж гри, користувач зможе досліджувати локацію, шукати тайні шляхи, або сховища, вступати в бій з ворогами та знищувати їх після чого збирати з них предмети (набої, аптечки та інше) [1, С.56].

2.2 Опис сценаріїв використання

Сценарій використання, або короткий сценарій, показує, як користувач взаємодіє з технологією в реальному світі. Він описує події, кроки та/або дії, які відбуваються під час контакту. Сценарії використання можуть бути досить детальними, пояснюючи, як саме користувач взаємодіє з інтерфейсом користувача, або вони можуть бути досить високого рівня, описуючі важливі бізнес-завдання, але не те, як вони виконуються.

В цьому проєкті, основна роль буде надана безпосередньо користувачу, який буде взаємодіяти з системою. Система виконує свої завдання на програмному рівні та контролює сам процес роботи застосунку. На таблицях 2.1-2.6 можна побачити приклади сценаріїв використання системи.

Таблиця 2.1 – Сценарій №1: Створення нового персонажу

Діючі особи	Користувач, система
Мета	Створення нового персонажу
Передумова	Користувач раніше не грав
Успішний сценарій: 1. Користувач знаходиться у меню створення гри. 2. Користувачу надається вибір складності гри. 3. Відкривається меню створення нового персонажу. 4. Система зберігає дані які заповнив та обрав користувач.	
Результат	Створення нового персонажу

Таблиця 2.2 – Сценарій №2: Генерація звичайного ігрового світу

Діючі особи	Користувач, система
Мета	Генерація звичайного ігрового світу
Передумова	Користувач вже створив ігрового персонажа
Успішний сценарій: 1. Користувач знаходиться у меню створення гри. 2. Користувачу надається продовжити грати обраним персонажем, або грати у режимі «Виживання».	

Кінець таблиці 2.2

3. Користувач обирає продовжити грати обраним персонажем, після чого система починає генерувати перший рівень. 4. Після генерації, користувачу буде надано можливість пройти перший рівень разом з підказками, це і буде його навчання.	
Результат	Генерація звичайного ігрового світу

Таблиця 2.3 – Сценарій №3: Генерація ігрового світу «Виживання»

Діючі особи	Користувач, система
Мета	Генерація ігрового світу «Виживання»
Передумова	Користувач вже створив ігрового персонажа
Успішний сценарій: 1. Користувач знаходиться у меню створення гри. 2. Користувачу надається продовжити грати обраним персонажем, або грати у режимі «Виживання». 3. Користувач обирає продовжити грати у режимі «Виживання», далі він має право обрати локацію на якій буде грати, або вибрати її випадково після чого система починає генерувати перший рівень. 4. Після генерації, користувачу буде надано можливість ввімкнути/вимкнути підказки.	
Результат	Генерація ігрового світу «Виживання»

Таблиця 2.4 – Сценарій №4: Покращення ігрового персонажу та спорядження

Діючі особи	Користувач, система
Мета	Покращення ігрового персонажу та амуніції
Передумова	Користувач вже грав та пройшов як найменше два рівні
Успішний сценарій: 1. Після завершення другого рівня, буде користувачу стане доступне меню покращення персонажу та амуніції. 2. При переході у це меню, поетапно будуть з'являтись підказки, що для чого залежить та як і що потрібно щоб це покращити. 3. Після усіх маніпулювань, система зберігає дані. 4. Користувач може вийти на головне меню, або продовжити грати далі.	
Результат	Покращення ігрового персонажу та амуніції

4.3 Алгоритм роботи програмного забезпечення

Алгоритм роботи програмного забезпечення створюється з метою опису логіки та послідовності дій, які виконує програма для досягнення своєї функціональності та мети. Основна ціль алгоритму роботи програмного забезпечення полягає в тому, щоб організувати та керувати послідовністю операцій та процесів, що відбуваються у програмі, щоб вона могла ефективно функціонувати та виконувати заплановані завдання.

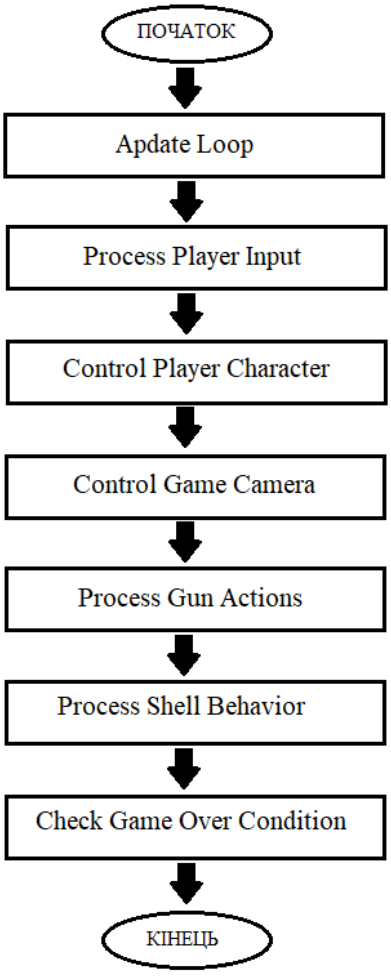

Рисунок 2.2 – Блок-схема роботи програмного забезпечення гри

Start Game. Початок гри. Ініціалізація необхідних об'єктів та налаштування початкових значень.

Update Loop. Основний цикл оновлення гри, що виконується кожен кадр. Включає в себе роботу з іншими блоками алгоритму.

Process Player Input. Обробка введення гравця. Зчитування натискання клавіш та взаємодія з грою.

Control Player Character. Керування персонажем гравця. Обробка руху гравця, поворотів та взаємодії з оточуючим середовищем.

Control Game Camera. Керування камерою гри. Налаштування позиції та орієнтації камери, щоб слідкувати за персонажем гравця.

Process Gun Actions. Обробка дій зі зброєю. Виявлення стрільби гравця та виконання відповідних дій, таких як постріл та відображення ефектів.

Process Shell Behavior. Обробка поведінки гільзи. Керування згасанням та зупинкою гільзи при зіткненні зі землею або іншими об'єктами.

Check Game Over Condition. Перевірка умови завершення гри. Перевірка, чи гра закінчилася, наприклад, через смерть гравця або досягнення певної мети.

End Game. Завершення гри. Виведення результатів, підрахунок очків або відображення інших відповідних повідомлень.

4.4 Діаграми станів та переходів

Діаграма станів та переходів є графічним зображенням станів, в яких може перебувати система або об'єкт, та переходів між цими станами. Вона використовується для моделювання та аналізу поведінки системи або об'єкта з точки зору його станів і переходів між ними.

Основна ціль створення діаграми станів та переходів полягає в тому, щоб візуалізувати та зрозуміти поведінку системи або об'єкта у вигляді послідовності станів та дій, які відбуваються в цих станах. Вона допомагає розкрити логіку роботи системи, виявити можливі стани та переходи між ними, а також виявити можливі проблеми або помилки в логіці системи.

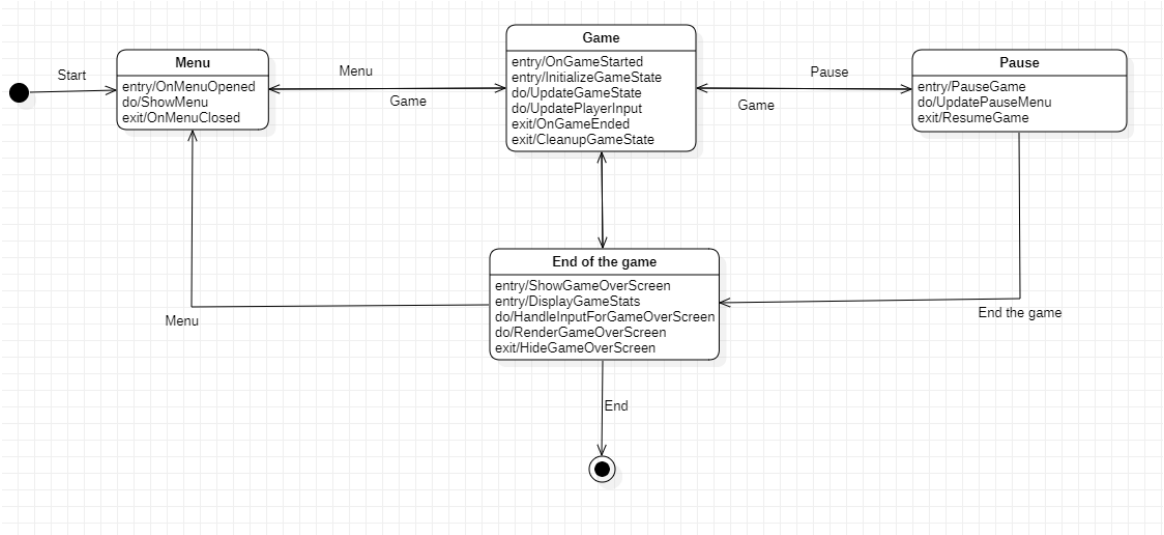

Рисунок 2.3 – Діаграма станів та переходів в цілому

OnMenuOpened - стан, у якому користувач перебуває на головному меню гри

OnGameOpened - стан, у якому користувач перебуває під час гри на рівні

Pause - стан, у якому гра зупинена, але не закінчена

ShowGameOverScreen - стан, у якому гра закінчена

InitializeGameState - перехід з меню в гру

PauseGame - перехід з гри в паузу

ResumeGame - перехід з паузи в гру

OnGameEnded - перехід з гри в кінець гри

ShowMenu - перехід з кінця гри в меню

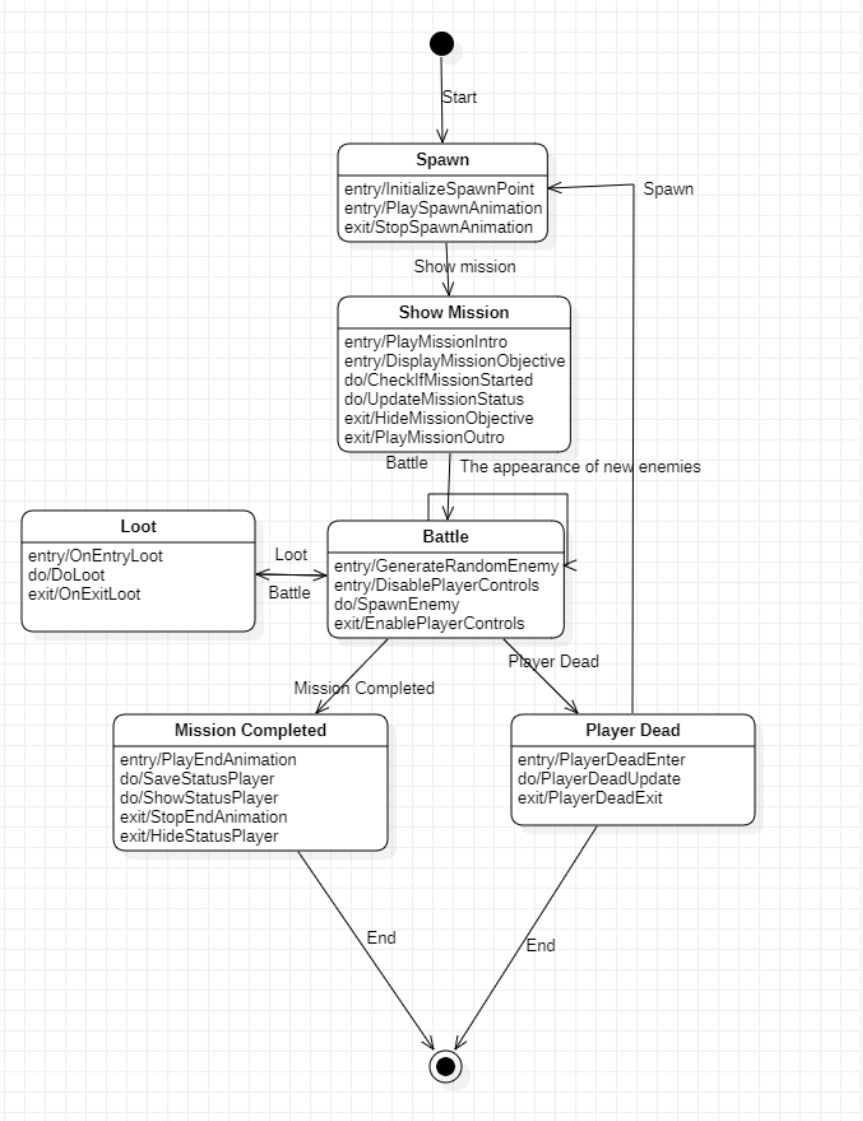

Рисунок 2.4 – Діаграма станів та переходів гемплею

Починається гра зі стану "Spawn", де гравець з'являється на екрані. Звідси можна перейти до стану "Show Mission", який показує завдання гравцеві. Зі стану "Show Mission" можна перейти до стану "Battle", де відбувається бій з ворогами. Якщо гравець переміг ворогів, то він переходить до стану "Loot", де він може зібрати різні предмети, такі як зброя, боєприпаси або аптечки. Зі стану "Loot" можна повернутися до стану "Battle", щоб продовжити бій з ворогами. Якщо гравець зібрав всі необхідні предмети і успішно виконав завдання, то він переходить до стану "Mission Completed". Якщо гравець помирає в бою, то він переходить до стану "Player Dead", звідки можна повернутися до стану "Spawn" для початку гри знову.

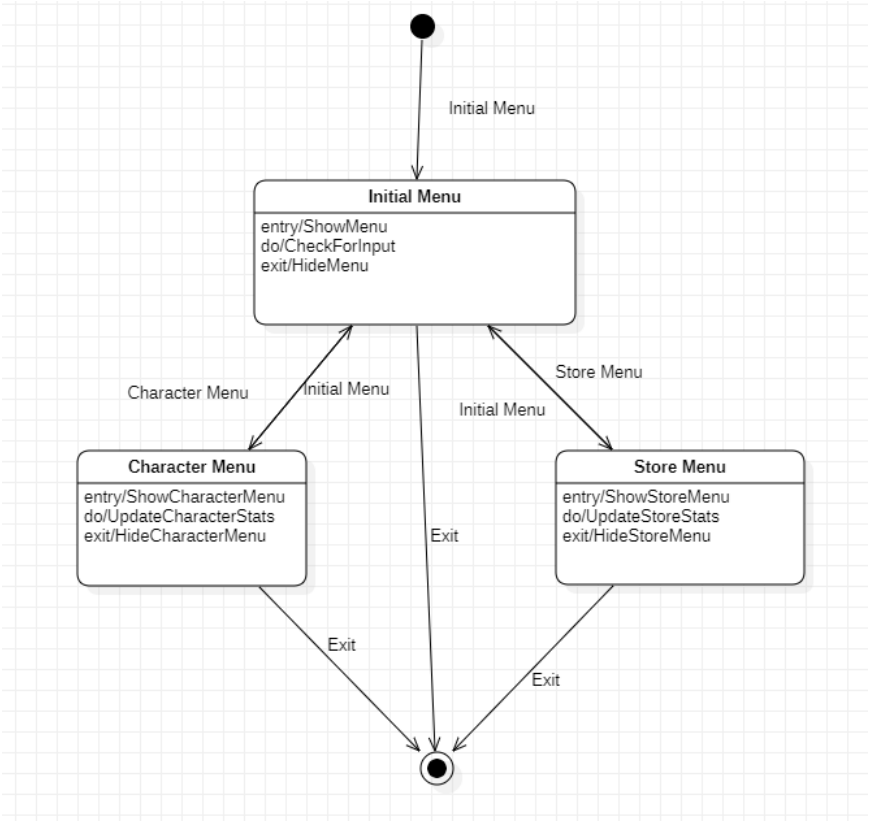

Рисунок 2.5 – Діаграма станів та переходів покращення характеристик

Початковим станом є "Initial Menu", після чого гравець може обрати один з трьох варіантів: "Character Menu", "Store Menu" або "Exit". Якщо гравець вибирає " Character Menu ", він може покращити свого персонажа, натиснувши на кнопку "Апгрейд персонажа", або повернутися назад до меню вибору виходу або переходу в меню магазину.

Якщо гравець вибирає " Store Menu ", він може вибрати, який тип зброї або припасу він хоче придбати, натиснувши на кнопку "Купити зброю" або "Купити припаси". Якщо гравець має достатньо грошей, щоб придбати зброю або припаси, він може купити їх та покращити своє зброю або запаси, інакше він повернеться до меню вибору виходу або переходу до меню персонажу.

У будь-який момент гравець може вибрати " Exit " та повернутися до початкового екрану гри.

2.5 Побудова діаграм класів

Діаграма класів – це візуальний засіб моделювання структури та взаємозв'язків між класами у програмному забезпеченні. Вона використовується для ілюстрації класів, їх атрибутів (змінних) та методів, а також зв'язків між класами.

Головна мета створення діаграми класів полягає в тому, щоб легше розуміти та візуалізувати архітектуру програми, структуру класів та залежності між ними. Вона надає загальний огляд класів і допомагає розробникам зрозуміти, як класи взаємодіють між собою та як вони організовані в системі.

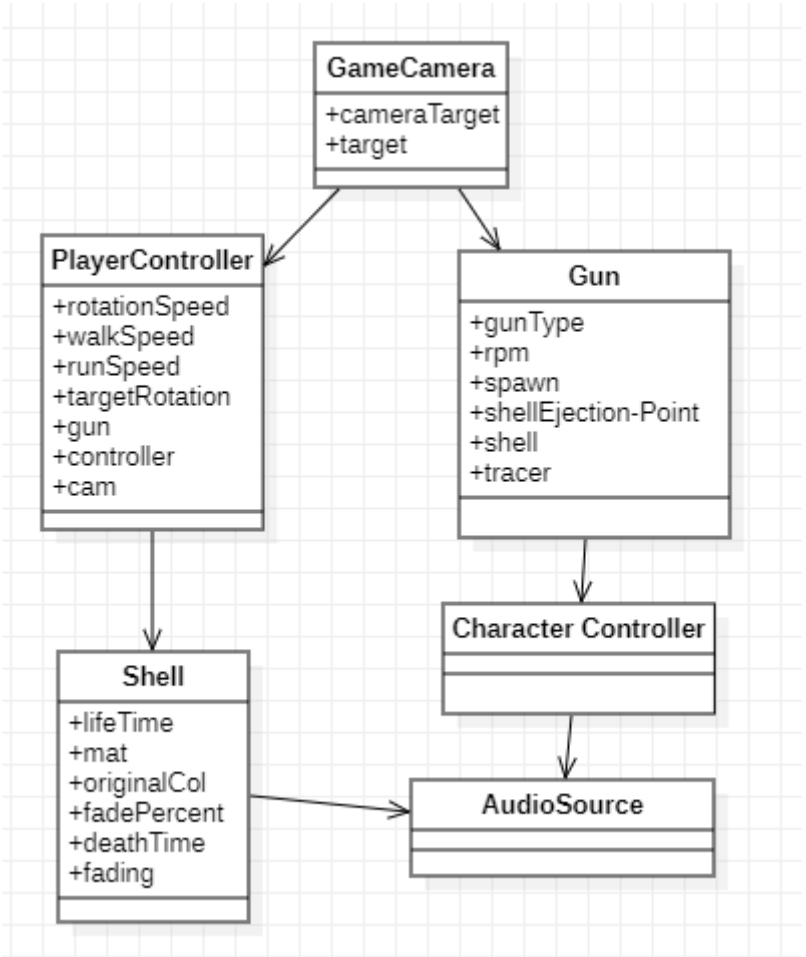

Рисунок 3.5 – Діаграма класів

GameCamera. Клас, який відповідає за рух камери в грі. Він використовується для слідкування за гравцем. Має поле cameraTarget для

зберігання позиції, на яку має спрямовуватися камера, і поле target, яке представляє об'єкт гравця.

PlayerController. Клас, що управляє гравцем. Містить поля і методи для обробки вводу гравця, руху та взаємодії зі зброєю. Використовує компонент CharacterController для керування рухом гравця та має посилання на камеру cam та зброю gun.

Gun. Клас, який представляє зброю гравця. Має поля, які визначають тип зброї та параметри (наприклад, кількість пострілів на хвилину). Використовується для стрільби та відтворення звуків пострілу. Містить посилання на об'єкт spawn для спавну кулі, shellEjectionPoint для спавну гільз та tracer для відображення лінії політу кулі.

Shell. Клас, що представляє гільзу. Відповідає за згасання гільзи з часом та зупинку гільзи при зіткненні зі землею. Містить логіку зміни кольору гільзи з часом та перевірку на зіткнення з об'єктами.

Висновки до розділу 2

У розділі 2 проведено проєктування гри засобами мови UML. Створено діаграму варіантів використання. Разрублено алгоритм роботи мобільної гри в жанрі Action на рушії Unity.

Описано сценарії використання гри, що розробляється. Наведено діаграми станів, які визначили та надали свого роду структурності системі, алгоритм роботи програмного забезпечення гри, їх атрибутів та взаємозв’язок взагалі.

На основі другого розділу можна починати розробку ігрового застосунку в жанрі Action.

3 КОДУВАННЯ МОБІЛЬНОЇ ГРИ В ЖАНРІ ACTION

3.1 Дослідження ігрових рушіїв

Перед тим як зробити вибір у платформи для розробки гри, важливо ретельно розглянути доступні рушії та їх характеристики:

Unreal Engine. Unreal Engine є сильним конкурентом Unity та відомий своїм потужним графічним движком. Він надає велику вагу на реалістичність графіки та деталізацію. Unreal Engine також використовує мову програмування C++, що дозволяє розробникам мати більш прямий контроль над процесом розробки. Крім того, Unreal Engine має велику спільноту та широкий набір інструментів для створення графічних ефектів та фізичної симуляції. Однак, варто зазначити, що Unreal Engine може вимагати більше обсягу ресурсів та може знадобитися більше часу та зусиль для вивчення його функціоналу та інструментів.

Рисунок 3.1 – Інтерфейс рушія Unreal Engine

Godot Engine. Godot Engine є іншим варіантом ігрового рушія для розробки ігор. Він відомий своєю простотою використання та невеликим розміром завантаження. Godot також має вбудовану мову програмування GDScript, яка є схожою до Python. Цей рушій пропонує інтуїтивний інтерфейс

та широкий спектр інструментів для розробки, включаючи візуальний редактор сцен та анімацій. Водночас, в порівнянні з Unity, Godot може бути менш потужним та обмеженим у певних аспектах, особливо для складних ігрових проектів.

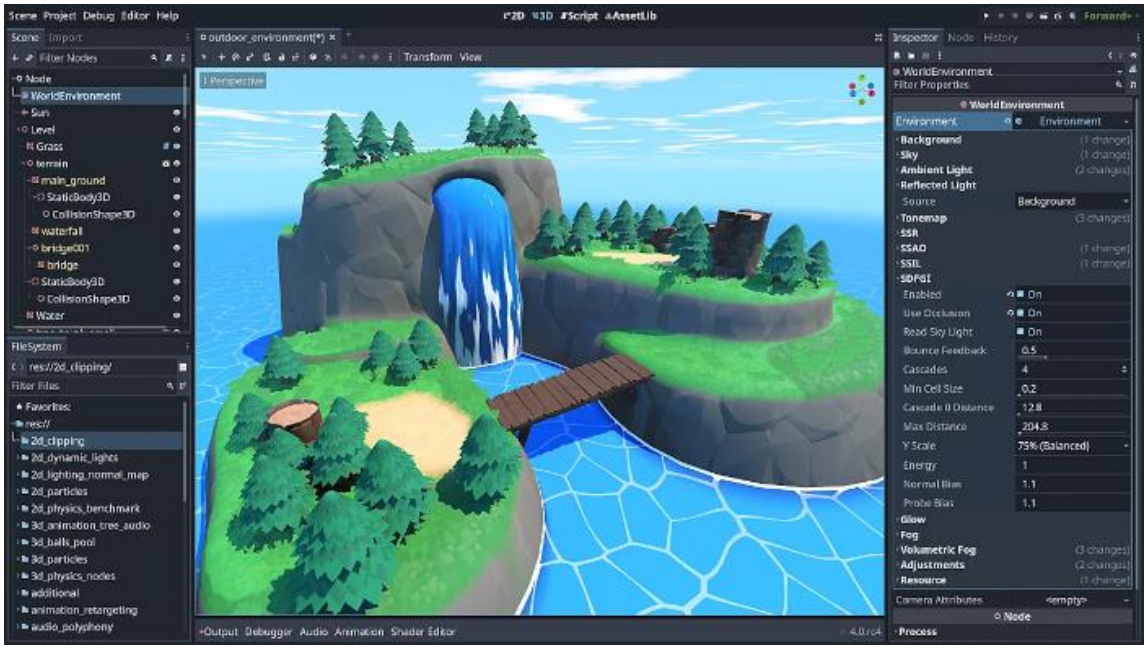

Рисунок 3.2 – Інтерфейс рушія Godot Engine

CryEngine. CryEngine є ще одним конкурентом Unity, відомим своїм фотореалістичним рендерингом та вражаючою графікою. Він пропонує потужні інструменти для створення відкритих світів та широкий набір ефектів, таких як динамічна освітлення та реалістична фізика. CryEngine також надає доступ до готових моделей та матеріалів, що може прискорити процес розробки. Однак, CryEngine може бути складним для новачків та вимагати більше обсягу ресурсів для ефективної роботи.

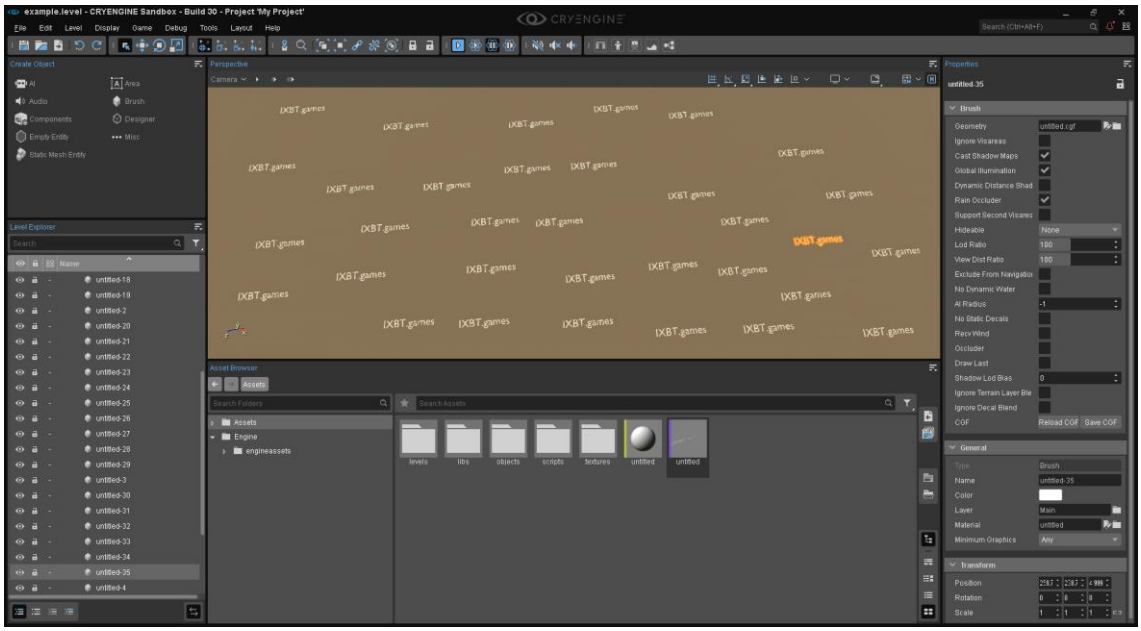

Рисунок 3.3 – Інтерфейс рушія CryEngine

Оглядаючи ці рушії, було зроблено вибір на користь Unity з таких причин:

Популярність та широке застосування: Unity є однією з найпопулярніших та найбільш використовуваних платформ для розробки ігор у світі. Вона використовується як незалежними розробниками, так і великими студіями, що забезпечує наявність широкої спільноти та велику кількість ресурсів для підтримки та навчання.

Кросплатформеність та розширення можливостей: Unity надає зручні інструменти для розробки ігор для різних платформ, включаючи комп'ютери, консолі, мобільні пристрої та віртуальну реальність. Це дозволяє досягати більшої аудиторії та забезпечує гнучкість у розповсюдженні гри. Крім того, Unity пропонує можливості для розширення за допомогою плагінів та сторонніх бібліотек, що дозволяє реалізувати різноманітні ідеї та функціональність.

Зручний інтерфейс та швидкість розробки: Unity має інтуїтивний інтерфейс та широкий набір інструментів, які допомагають розробникам швидко перетворювати свої ідеї в реальність. Крім того, наявність мови C# сприяє швидкому розробленню логіки гри та забезпечує ефективність роботи.

Unity пропонує потужний високорівневий редактор, який дозволяє розробникам легко створювати та редагувати графічний вміст. Деякі з переваг цього редактора включають:

Інтуїтивний і зручний інтерфейс. Unity забезпечує зрозумілий та легкий у використанні інтерфейс, що дозволяє розробникам швидко орієнтуватися та працювати з графічним вмістом;

Вбудовані інструменти створення. Unity надає широкий спектр вбудованих інструментів для створення графічного вмісту, таких як моделювання 3D-об'єктів, текстурування, анімація, освітлення та багато іншого;

Підтримка імпорту різноманітних форматів. Unity підтримує імпорт графічного вмісту у різних форматах, включаючи 3D-моделі, текстури, звуки, анімації та інші, що робить його універсальним інструментом для роботи з різними типами контенту.

Unity має велику та активну спільноту розробників, що є важливим перевагою для користувачів цього движка. Ось деякі переваги великої спільноти та екосистеми Unity:

Підтримка та документація. Завдяки великій спільноті розробників, ви зможете знайти відповіді на свої питання, рішення для проблем та корисні поради у форумах, блогах, соціальних медіа та офіційній документації Unity;

Навчальні матеріали та онлайн-курси. Багато розробників та освітніх установ надають онлайн-курси, відеоуроки та навчальні матеріали для вивчення Unity. Це дозволяє вам швидко освоїти нові можливості та покращити навички розробки;

Спільнотні ресурси. Unity має широкий спектр спільнотних ресурсів, таких як безкоштовні асети, плагіни та скрипти, які розробники можуть використовувати для прискорення своєї роботи та поліпшення якості своїх ігор.

Незважаючи на багато переваг Unity, варто зазначити кілька обмежень та недоліків, які можуть вплинути на розробку ігор:

Швидкодія. Деякі розробники вказують на те, що Unity може мати певні обмеження щодо швидкодії, особливо для великих та складних проектів. Оптимізація коду та використання правильних методик можуть допомогти у подоланні цього обмеження;

Обмежені можливості графічного движка. В порівнянні з деякими іншими графічними движками Unity може мати обмежені можливості у сфері графіки. Проте, використання спеціалізованих плагінів та налаштування можуть допомогти забезпечити високу якість графічного візуалу.

3.2 Інтеграція мови програмування C# та необхідних бібліотек

Однією з головних переваг використання мови програмування C# у розробці гри з використанням Unity є його потужна екосистема, що робить його привабливим для багатьох розробників. Переваги C# в контексті Unity включають:

Простота вивчення. C# має схожий синтаксис з іншими мовами програмування, такими як Java або C++, що полегшує вивчення мови розробниками з досвідом у цих мовах;

Інтеграція з Unity. C# є офіційною мовою сценаріїв для Unity, що означає, що він має вбудовану підтримку та найбільшу функціональність для розробки у цьому середовищі;

Широкі можливості. C# надає доступ до багатьох можливостей Unity, таких як робота з графічними об'єктами, фізичними ефектами, звуком, штучним інтелектом та іншими системами, що розширюють можливості розробки гри;

Ефективність та продуктивність. C# дозволяє досягти високої продуктивності в грі, забезпечуючи швидку обробку даних та ефективне використання ресурсів.

Unity надає потужну інтеграцію для мови C#, що дозволяє розробникам використовувати багато функцій цього середовища. Деякі ключові аспекти інтеграції C# з Unity включають:

Unity API. Unity надає широкий набір API та класів, які дозволяють вам керувати об'єктами, фізикою, анімацією, звуком та багатьма іншими аспектами гри. Ці класи доступні для використання в C# скриптах;

Можливості компонентів. Компонентна модель Unity дозволяє використовувати скрипти C# як компоненти, що можуть бути прикріплені до гральних об'єктів. Це робить можливим додавання функціональності до об'єктів гри та контроль їхньої поведінки;

Події та делегати. C# має підтримку подій та делегатів, що дозволяють зручно реагувати на події в грі, такі як зіткнення об'єктів або кліки гравця. Це спрощує розробку систем взаємодії та обробку різноманітних подій.

При розробці гри в Unity з використанням C#, може бути необхідно встановити деякі додаткові бібліотеки або залежності для розширення функціональності або полегшення розробки. Деякі основні кроки, які можна виконати для встановлення необхідних бібліотек, включають:

Використання пакетного менеджера. Unity має вбудований пакетний менеджер, який дозволяє швидко встановлювати бібліотеки з Unity Asset Store або зовнішніх джерел. Це полегшує процес встановлення необхідних розширень;

Вручну встановлення бібліотек. Якщо необхідна бібліотека не доступна через пакетний менеджер Unity, можливо, вам доведеться встановити її вручну. Це може включати завантаження джерела коду, компіляцію та додавання скомпільованої бібліотеки до вашого проекту Unity;

Керування залежностями. У деяких випадках, для розробки в Unity може знадобитися керування залежностями між різними бібліотеками або модулями. Ви можете використовувати інструменти керування залежностями,

такі як NuGet або Unity Package Manager, для управління залежностями вашого проекту.

У процесі розробки необхідно було накладати матеріали, але через малу кількість яку надають спочатку не було можливості накласти необхідні. Постало питання як же потрібно додавати нові необхідні пакети або бібліотеки до Unity?

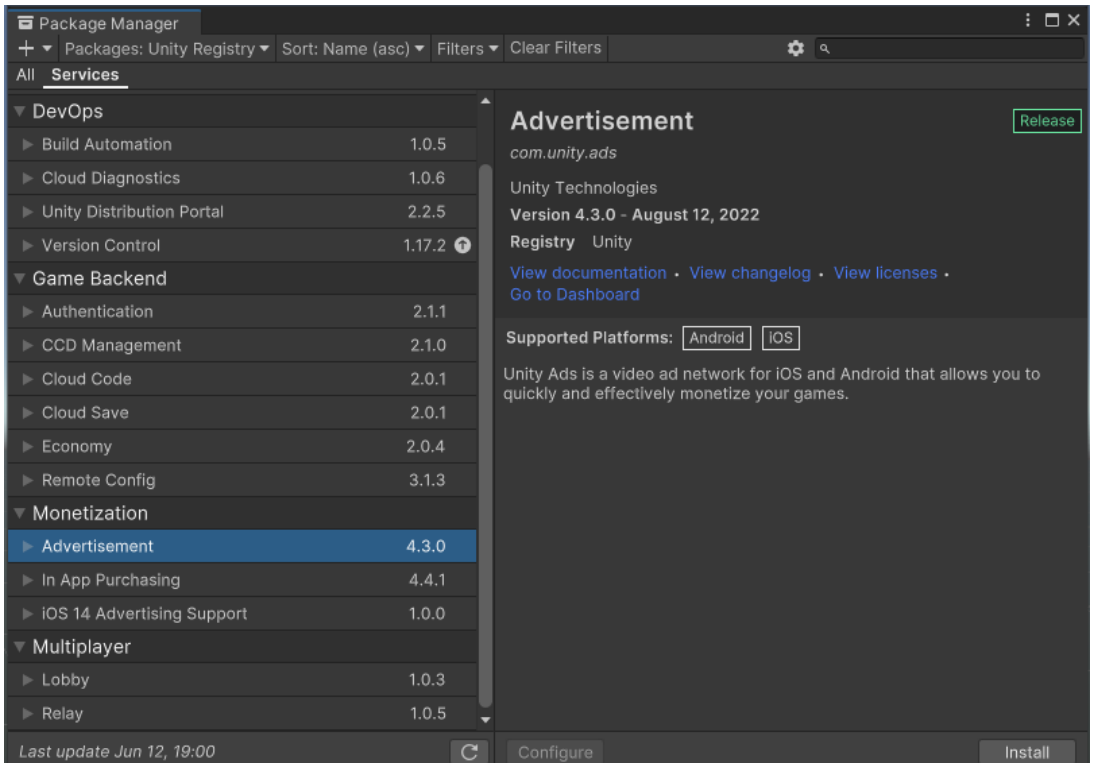

Рисунок 3.4 – Вікно “Package Manager”

Для додавання як старих та і нових бібліотек в Unity Package Manager потрібно виконати наступні кроки:

У Package Manager ви побачите список доступних пакетів. Прокрутіть список вниз, доки не побачите кнопку "Add package from disk" (Додати пакет з диска) в правому верхньому куті вікна Package Manager. Натисніть на цю кнопку.

У вікні "Add package from disk" виберіть шлях до старої бібліотеки, яку ви хочете додати. Це може бути файл .unitypackage або папка з бібліотекою Unity.

Після вибору файлу або папки натисніть "Open" (Відкрити), і Unity почне імпортувати бібліотеку.

Після завершення імпорту бібліотеки вона з'явиться у списку доступних пакетів Package Manager. Ви можете встановити, видалити або оновити цей пакет за допомогою відповідних кнопок.

Це спосіб додавання старих бібліотек в Unity Package Manager. Ви можете використовувати цей метод для управління своїми бібліотеками та розширеннями в проектах Unity.

3.3 Розробка скриптів

Огляд створених скриптів забезпечує розуміння основної функціональності та взаємодії компонентів в грі, розробленої на платформі Unity. Розглянемо головні скрипти, які були створені для реалізації руху гравця, логіки зброї, керування камерою та інших нюансів гри. Кожен з цих скриптів відповідає за конкретну частину функціональності і взаємодіє з іншими компонентами гри, створюючи змістовний і цілісний геймплей. Розглянемо кожен скрипт окремо та визначимо його роль і внесок у загальну механіку гри.

Перший і найголовніший це скрипт "GameCamera"(додаток А), бо які б складні механіки ми б не робили, все це буде безглуздо, якщо користувач не зможе їх фізично побачити. Отож, головна мета скрипта "GameCamera" полягає у відслідковуванні руху гравця і руху камери, щоб забезпечити плавний перехід та наведення камери на гравця. Основні елементи цього скрипта включають:

Директиви using. У цьому скрипті використовується директива using System.Collections для використання колекцій, а також using UnityEngine для доступу до класів та функцій, пов'язаних з Unity.

Змінні. Скрипт має приватну змінну cameraTarget, яка зберігає позицію, на яку спрямовується камера. Також є приватна змінна target, яка зберігає посилання на об'єкт гравця.

Метод Start(). У методі Start() скрипт отримує посилання на об'єкт гравця за допомогою методу GameObject.FindGameObjectWithTag(). Він шукає об'єкт з тегом "Player" і зберігає його трансформацію у змінній target.

Метод Update(). У методі Update() відбувається оновлення позиції камери. Змінна cameraTarget оновлюється, щоб мати позицію гравця зі зміненою висотою (тримаємо Y-координату камери незмінною). Потім застосовується метод Vector3.Lerp(), який забезпечує плавний перехід між поточною позицією камери та цільовою позицією (cameraTarget). Time.deltaTime * 8 використовується для контролю швидкості переходу камери.

Далі йде скрипт "PlayerController"(додаток В). Головна мета скрипта полягає у відслідковуванні руху гравця і руху камери, щоб забезпечити плавний перехід та наведення камери на гравця. Основні елементи цього скрипта включають:

Директиви using. У скрипті використовується директива using System.Collections для використання колекцій, а також using UnityEngine для доступу до класів та функцій, пов'язаних з Unity.

Атрибут [RequireComponent]. Цей атрибут застосовується до класу і вказує, що об'єкт, на якому цей скрипт прикріплений, повинен мати компонент CharacterController. Це забезпечує, що необхідний компонент присутній.

Змінні. У скрипті є декілька змінних, таких як rotationSpeed, walkSpeed та runSpeed, які визначають параметри руху гравця і швидкість обертання. Є також змінні для зберігання об'єктів компонентів, таких як gun (для взаємодії зі скриптом зброї), controller (для доступу до CharacterController) та cam (для доступу до головної камери).

Метод Start(). У методі Start() отримується посилання на CharacterController і головну камеру, використовуючи GetComponent() та Camera.main відповідно.

Метод Update(). У методі Update() відбувається оновлення керування гравцем. Метод ControlMouse() відповідає за керування гравцем за допомогою миші, включаючи обертання гравця в напрямку позиції миші, визначення швидкості руху гравця та виклик методів зброї відповідно до взаємодії зі спусковою кнопкою. Метод ControlWASD() відповідає за керування гравцем за допомогою клавіш WASD, включаючи обертання гравця в напрямку натискання клавіш, визначення швидкості руху гравця та виклик методів зброї.

Методи Move і LookRotation. В обох методах ControlMouse() та ControlWASD() використовуються методи для зміщення гравця (controller.Move()) та обертання гравця (transform.eulerAngles), забезпечуючи плавний рух і обертання залежно від введених команд гравцем.

Далі ми маємо скрипт "Gun"(додаток Б), який відповідає за функціональність зброї гравця. Основні елементи цього скрипта включають:

Директиви using. У скрипті використовується директива using System.Collections для використання колекцій, а також using UnityEngine для доступу до класів та функцій, пов'язаних з Unity.

Атрибут [RequireComponent]. Цей атрибут застосовується до класу і вказує, що об'єкт, на якому цей скрипт прикріплений, повинен мати компонент AudioSource. Це забезпечує, що необхідний компонент присутній.

Змінні. У скрипті є декілька змінних, таких як gunType (тип зброї), rpm (кількість пострілів на хвилину), spawn (позиція викидання куль), shellEjectionPoint (позиція викидання гільз), shell (префаб гільзи), tracer (об'єкт LineRenderer для візуалізації траєкторії пострілу) та деякі додаткові змінні для системи розрахунку часу пострілу.

Метод Start(). У методі Start() ініціалізуються змінні для розрахунку часу між пострілами на основі значення rpm. Також перевіряється наявність компонента LineRenderer і, якщо він присутній, зберігає посилання на нього в змінну tracer.

Метод Shoot(). Цей метод викликається при пострілі зброї. Він перевіряє, чи можливо виконати постріл на даний момент (час минув від останнього пострілу), створює промінь (ray) з позиції викидання (spawn) у напрямку вперед, перевіряє, чи перетинає промінь який-небудь об'єкт, обчислює відстань пострілу, оновлює час наступного можливого пострілу, відтворює звук пострілу та ініціює випуск гільзи з відповідною силою і напрямком.

Метод ShootContinuous(). Цей метод викликається при тривалому утриманні кнопки пострілу. Якщо тип зброї gunType встановлений на Auto, виконується метод Shoot().

Приватний метод CanShoot(). Цей метод перевіряє, чи можна здійснити постріл на даному етапі гри, перевіряючи час (чи пройшов достатній час з моменту останнього пострілу).

Корутин RenderTracer(). Цей корутин використовується для візуалізації траєкторії пострілу. Він активує компонент LineRenderer, встановлює початкову та кінцеву позицію для візуалізації траєкторії, а потім вимикає компонент після короткого затримки.

Також ми маємо скрипт "Shell"(додаток А), який відповідає за поведінку гільзи після викидання. Основні елементи цього скрипта включають:

Директиви using. У скрипті використовується директива using System.Collections для використання колекцій, а також using UnityEngine для доступу до класів та функцій, пов'язаних з Unity.

Змінні. У скрипті є декілька змінних, таких як lifeTime (тривалість життя гільзи), mat (матеріал гільзи), originalCol (початковий колір гільзи), fadePercent

(відсоток згасання колір гільзи), deathTime (час, коли гільза починає згасати), fading (прапорець, що показує, чи гільза згасає).

Метод Start(). У методі Start() ініціалізуються змінні, такі як матеріал гільзи, початковий колір, час згасання і запускається корутин Fade.

Корутин Fade(). Цей корутин використовується для згасання гільзи з часом. У циклі корутини, кожні 0.2 секунди перевіряється, чи гільза знаходиться в стані згасання. Якщо так, то збільшується fadePercent (відсоток згасання), змінюється колір гільзи до прозорого (Color.Lerp), і якщо fadePercent досягає або перевищує 1, гільза знищується. Якщо гільза не знаходиться в стані згасання, перевіряється, чи пройшов час deathTime, якщо так, то змінна fading встановлюється в true і гільза починає згасати.

Метод OnTriggerEnter(). Цей метод викликається, коли гільза зіштовхується з коллайдером. Якщо тег коллайдера є "Ground" (земля), то гільза зупиняється за допомогою Sleep().

3.4 Архітектура та організація скриптів

Зважаючи на архітектуру та організацію наших скриптів, ми можемо виділити кілька ключових аспектів, які сприяють ефективності та модульності нашого коду.

Використання об'єктно-орієнтованого підходу при написанні скриптів дозволяє нам організувати код вокруг об'єктів, які взаємодіють між собою. Кожен скрипт може відповідати за конкретний об'єкт або його частину, і ми можемо використовувати наслідування, композицію та інші принципи ООП для забезпечення чистоти і структурованості коду.

Модульна структура та розділення логіки на окремі скрипти допомагають нам зберігати код організованим та легко зрозумілим. Кожен скрипт відповідає за конкретний аспект гри, такий як рух гравця, логіка зброї чи керування камерою. Це дозволяє нам зосередитися на конкретній функціональності та забезпечує легкість розширення та підтримки коду.

Використання делегатів, подій та інших концепцій C# для зв'язку скриптів дозволяє нам створювати гнучку систему взаємодії між компонентами гри. Наприклад, ми можемо використовувати делегати для передачі функцій між скриптами і викликати їх у відповідних моментах. Також, використання подій дозволяє нам сповіщати інші скрипти про відбуття певних подій в грі, що спрощує взаємодію та реалізацію різноманітних функціональних залежностей.

Завдяки такій архітектурі та організації наших скриптів, ми отримуємо гнучку та легко розширювану систему, де кожен скрипт виконує конкретну функцію і забезпечує взаємодію з іншими компонентами гри.

3.5 Стандартні пакети Unity

Основні стандартні пакети Unity надають широкий набір функціональності для розробки ігор. Ось кілька основних пакетів, які можуна розглянути:

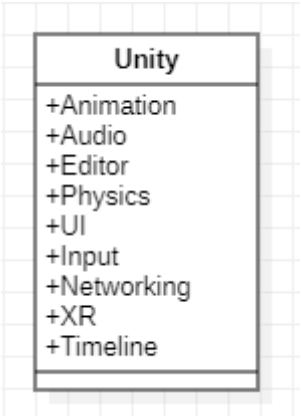

Рисунок 3.5 – Основні пакети Unity

Unity. Основний пакет Unity, який містить основну функціональність двигуна гри.

Animation. Пакет, що надає інструменти для роботи з анімаціями персонажів та об'єктів в грі.

Audio. Пакет, що містить інструменти для роботи зі звуком, включаючи відтворення звуків, музики та звукові ефекти.

Editor. Пакет, який надає інструменти для розробки редактора Unity та налаштування робочого середовища.

Physics. Пакет, що надає фізичну симуляцію та реалістичну поведінку об'єктів у грі.

UI. Пакет, який містить інструменти для створення інтерфейсу користувача, включаючи кнопки, тексти, панелі та інші елементи.

Input. Пакет, що надає можливості для обробки вводу користувача, такі як клавіатура, миша та контролери.

Networking. Пакет, який надає функціональність для мережевого взаємодії між гравцями, включаючи локальну та віддалену гру.

XR. Пакет, що надає підтримку розширеної реальності (AR) та віртуальної реальності (VR), включаючи різні пристрої та платформи.

Timeline. Пакет, що надає інструменти для створення та керування сценаріями та анімаціями у грі.

Зазначені пакети представляють лише деякі з основних стандартних пакетів, доступних у Unity. Залежно від конкретних потреб проекту, можуть бути включені інші пакети або сторонні розширення.

3.6 Діаграма компонентів

Діаграма компонентів - це візуальний інструмент, який демонструє структуру гри за допомогою компонентів, які присутні в кожному об'єкті. Нижче наведена діаграма компонентів:

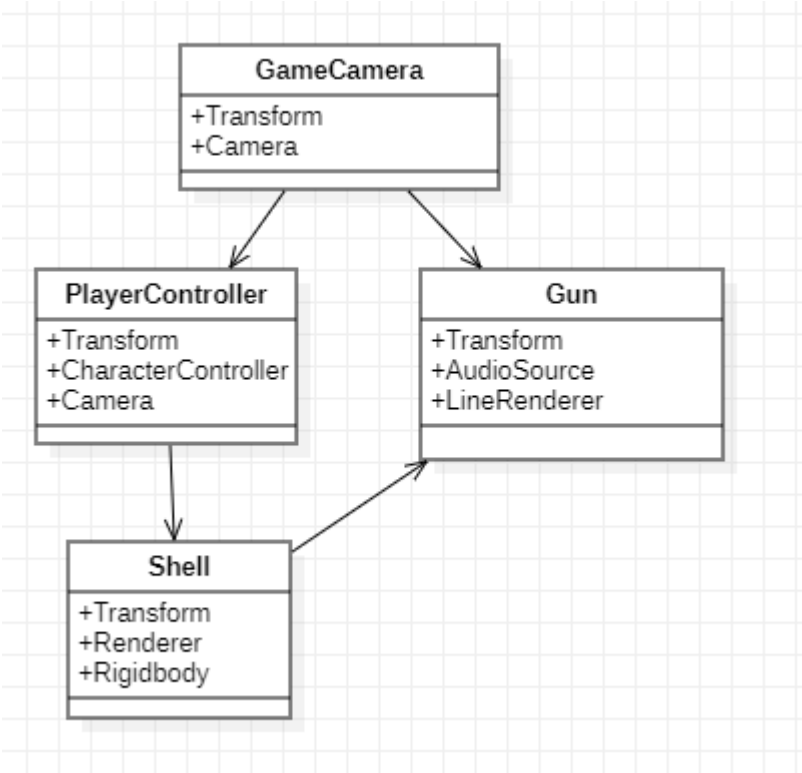

Рисунок 3.6 – Діаграма компонентів

GameCamera. Компонент, що дозволяє керувати камерою в грі. Використовує компоненти Transform та Camera для налаштування позиції та властивостей камери.

PlayerController. Компонент, який відповідає за управління гравцем. Включає в себе компоненти Transform, CharacterController для керування рухом гравця та Camera для налаштування погляду.

Gun. Компонент, що представляє зброю гравця. Включає в себе компоненти Transform для позиціонування зброї, AudioSource для відтворення звуків та LineRenderer для відображення лінії політу кулі.

Shell. Компонент, який представляє гільзу. Містить компоненти Transform для позиціонування гільзи, Renderer для відображення графіки та Rigidbody для обробки фізики гільзи.

3.7 Огляд створеного застосунку

Всі моделі були розроблені у програмі для 3D скульптингу ZBrush, у подальшому були імпортовані для текстурування, додавання кісток та текстурування.

Рисунок 3.7 – Модель головного персонажу

Головний персонаж являє собою солдата з амуніцією для штурмування закритих територій. У подальшому йому буде надано стандартна зброя, де користувач зможе вибирати режим стрільби: одиночний, бурст та автоматичний. Це зроблено для різних ситуацій чи економлення припасів, бо набої не нескінченні. Також було додано трасер після пострілу, після пострілу може з'явитися трасер, це зроблено для того щоб користувач міг легше коригувати свій вогонь по ворогам.

Рисунок 3.8 – Модель ворога

Ворог являє собої мародера-мутанта. Основним їх завданням стане слідкуванням за конкретною територією. Даний ворог матиме прописаний огляд з полем в 90^{o} і якщо він побачить головного героя тоді зреагує трігер та він почне атакувати його. На початку це будуть рукопашний бій, а на рівнях вище він почне викоистовувати підручні матеріали(холодну зброю) та надані їх вогнестрільну зброю. Можливо у подальшому їм буде надано ще більш ширше озброєння та амуніцію, наприклад: гранати, аптечки та інше.

Рисунок 3.9 – Модель ігрового рівня

Ігровий рівень, або «карта» була спроектувана у програмі для моделювання та скульптингу Blender. Було вирішено не робити якусь одну локацюю, а зробити кімнати та у подальшому їх з'єднувати, це надасть змогу робити більш гнучкі та різноманітні рівні.

Рисунок 3.10 – Анімація руху головного персонажу

Анімація руху буде додана одна, бо загалом немає сенсу їх додавати, бо за героєм користувач буде наглядати зверху і не буде бачити анімації. Проте він побачить позу в якій рухається герой, також було додано анімацію пострілу, а саме при пострілі герой через віддачу героя почне смикати у той бік в якому знаходиться зброя. Причому це ще буде залежати від режиму пострілу, який обере користувач, тобто якщо це буде одиночний постріл це можна і не помітити, але якщо це буде автоматичний, тоді героя почне смикати настільки, що його може навіть повернути у бік

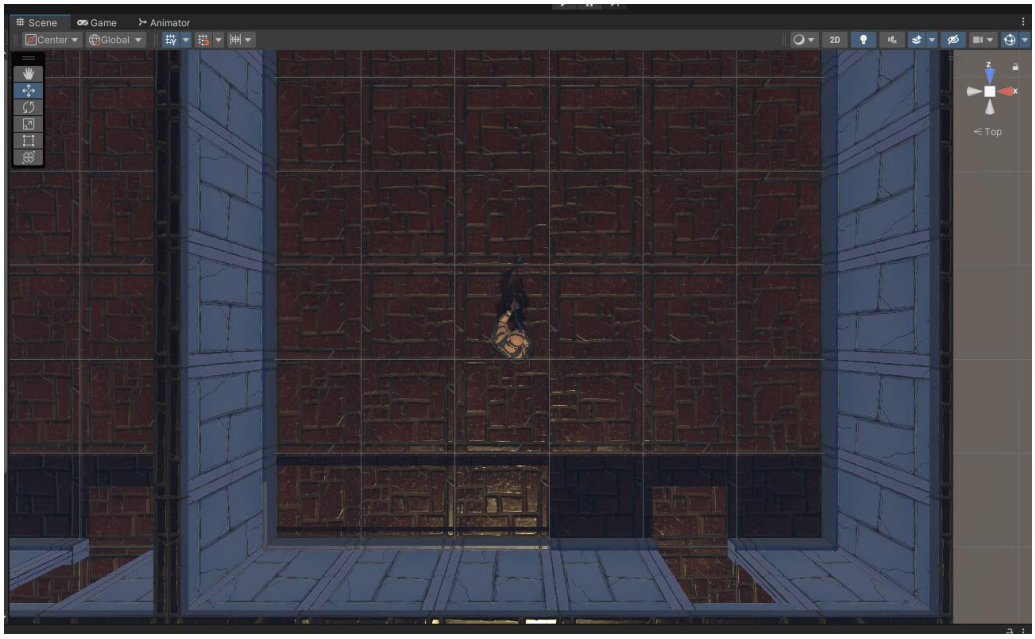

Рисунок 3.11 – Камера слідкування за головним героєм

Було вирішено зробити камеру слідкування над персонажем, це полегшить слідкування і користувач не загубить його, також користувачу буде легше досліджувати рівні, «підглядувати» за ворогами через кімнати, але не завжди, бо деякі кімнати будуть у тумані і розвіяти його муожна лише, якщо герой увійде в неї. Це запустить додаткове завдання, або це означатиме що у даній кімнаті є якась схованка з корисними припасами.

Висновки до розділу 3

У цьому розділі ми розглянули процес розвитку гри, використовуючи Unity та мову програмування C#. Обгрунтовано вибір Unity як потужного

інструменту для розробки ігор, а також переваги та недоліки використання Unity. Далі було детально розглянуто кожен з наших скриптів, які відповідають за рух гравця, логіку зброї, керування камерою та інші дрібниці, такі як гільзи та сліди польоту пулі.

Одним з ключових аспектів було використання об'єктно-орієнтованого підходу при написанні скриптів. Це дозволило організувати код навколо конкретних об'єктів та розділити логіку на окремі скрипти.

В цілому, розвиток гри з використанням Unity та мови C# виявився успішним завдяки ефективній архітектурі, модульній структурі та використанню різних концепцій програмування. Це дозволило нам створити функціональну та захоплюючу гру з різноманітними можливостями та спеціальними ефектами.

ВИСНОВКИ

У даній роботі проведено аналіз предметної галузі та розглянуті різні методи та види проектування програмних систем. Створено діаграми прецедентів, взаємодії та класів, а також діаграми станів та переходів. Розроблено діаграми діяльності, компонентів та розгортання, а також діаграми пакетів. В процесі роботи використано сучасні інструменти візуального моделювання, які дозволяють швидко та ефективно створювати складні системи.

Отримані результати показали, що використання методів та інструментів візуального моделювання є дуже ефективним та зручним способом проектування програмних систем. Вони дозволяють знизити час розробки та зменшити кількість помилок в процесі розробки.

Діаграми прецедентів та взаємодії допомагають зрозуміти функціональність системи та взаємодію з користувачем. В роботі наведено діаграму прецедентів. Описано алгоритм… Діаграми класів та компонентів дозволяють краще розуміти структуру системи та її модульність. Діаграми станів та переходів допомагають розібратись у поведінці системи та знайти можливі помилки.

Отже, використання візуального моделювання є важливим етапом в розробці програмних систем та дозволяє покращити якість та ефективність проектування.

Розроблене програмне забезпечення є грою, яка поєднує в собі функціональність, ефективність та реалістичний ігровий досвід, а використання візуального моделювання в процесі розробки дало можливість створити цю гру швидко та якісно.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Burgun K. Game design theory: a new philosophy for understanding games. Boca Raton, FL : A K Peters/CRC Press, 2012. p.188

2. Fullerton T. Game design workshop: a playcentric approach to creating innovative games. 2nd ed. Amsterdam : Elsevier Morgan Kaufmann, 2008. p. 496.

3. Hodent C. Gamer's brain: how neuroscience and UX can impact video game design. Taylor & Francis Group, 2017. p. 272.

4. Madhav S. Game programming algorithms and techniques: a platform-agnostic approach. Pearson Education, Limited. p. 352.

5. Nystrom R. Game programming patterns. Apress, 2011. P. 354.

6. Rogers S. Level up! The guide to great video game design. Wiley & Sons, Incorporated, John, 2014. P. 560.

7. Schell J. Art of game design: a book of lenses. Taylor & Francis Group, 2008. p. 520.

8. Swink S. Game feel: a game designer's guide to virtual sensation. Taylor & Francis Group, 2008. p. 376.

9. Construct 3 manual. офіційна документація Construct 3. URL: https://www.construct.net/en/make-games/manuals/construct-3 (дата звернення: 20.04.2023).

10. Gamasutra. веб-сайт, присвячений ігровій індустрії, зі статтями, новинами та інтерв'ю з ігровими розробниками. URL: https://www.gamasutra.com/ (дата звернення: 20.04.2023).

11. GameMaker studio 2 manual. офіційна документація GameMaker Studio 2, ще одного інструменту для створення ігор. URL: https://manual.yoyogames.com/ (дата звернення: 20.04.2023).

12. Godot engine. безкоштовний відкритий інструмент для розробки ігор, що підтримує як 2D, так і 3D-ігри. URL: https://godotengine.org/ (дата звернення: 20.04.2023).

13. Krita. безкоштовний відкритий програмний засіб для малювання та цифрової графіки. URL: https://krita.org/en/ (дата звернення: 20.04.2023).

14. The game developers conference vault. онлайн-бібліотека з презентаціями, відео та статтями, які були представлені на Game Developers Conference (GDC). URL: https://www.gdcvault.com/ (дата звернення: 20.04.2023).

15. The ultimate guide to game development with unity" by jonathan weinberger. платний онлайн-курс на платформі Udemy, але часто доступний зі знижкою, що надає введення в розробку ігор з використанням Unity. "Introduction to Unreal Engine" by Epic Games. URL: https://www.udemy.com/course/the-ultimate-guide-to-game-development-with-unity/ (дата звернення: 20.04.2023).

16. Unity learn. безкоштовна онлайн-платформа від Unity, що надає навчальні матеріали для розробки ігор, від початкового до продвинутого рівня. URL: https://learn.unity.com/ (дата звернення: 20.04.2023).

ДОДАТОК А

Скрипт «GameCamera»

```
void Start()
    {
        target = GameObject.FindGameObjectWithTag("Player").transform;
    }

    void Update()
    {
        cameraTarget = new
Vector3(target.position.x,transform.position.y,target.position.z);
        transform.position = Vector3.Lerp(transform.position,
cameraTarget,Time.deltaTime * 8);
```

Скрипт «Shell»

```
IEnumerator Fade()
    {
        while(true)
        {
            yield return new WaitForSeconds(.2f);
            if (fading)
            {
                fadePercent += Time.deltaTime;
                mat.color = Color.Lerp(originalCol, Color.clear, fadePercent);

                if (fadePercent >= 1)
                {
                    Destroy(gameObject);
                }
            }
            else
            {
                if(Time.time > deathTime)
                {
                    fading = true;
                }
            }
        }
    }

    private void OnTriggerEnter(Collider c)
    {
        if(c.tag == "Ground")
        {
            GetComponent<Rigidbody>().Sleep();
        }
          }
```

ДОДАТОК Б

Скрипт «Gun»

```
public void Shoot()
    {
        if (CanShoot())
        {
            Ray ray = new Ray(spawn.position, spawn.forward);
            RaycastHit hit;

            float shotDistance = 20;

            if (Physics.Raycast(ray, out hit, shotDistance))
            {
                shotDistance = hit.distance;
            }

            nextPossibleShootTime = Time.time + secondBetweenShots;

            GetComponent<AudioSource>().Play();

            if(tracer)
            {
                StartCoroutine("RenderTracer", ray.direction * shotDistance);
            }

            Rigidbody newShell = Instantiate(shell,
shellEjectionPoint.position,Quaternion.identity) as Rigidbody;
            newShell.AddForce(shellEjectionPoint.forward * Random.Range(150f, 200f)
+ spawn.forward * Random.Range(-10f, 10f));
        }
    }

    public void ShootContinuous()
    {
        if (gunType == GunType.Auto) {
            Shoot();
        }
    }

    private bool CanShoot()
    {
        bool canShoot = true;

        if (Time.time < nextPossibleShootTime)
        {
            canShoot = false;
        }

        return canShoot;
    }

    IEnumerator RenderTracer(Vector3 hitPoint)
    {
        tracer.enabled = true;
        tracer.SetPosition(0, spawn.position);
        tracer.SetPosition(1,spawn.position + hitPoint);

        yield return null;
        tracer.enabled = false;
          }
```

ДОДАТОК В

Скрипт «PlayerController»

```
void ControlMouse()
    {
        Vector3 mousePos = Input.mousePosition;
        mousePos = cam.ScreenToWorldPoint(new
Vector3(mousePos.x,mousePos.y,cam.transform.position.y - transform.position.y));
        targetRotation = Quaternion.LookRotation(mousePos - new
Vector3(transform.position.x,0,transform.position.z));
        transform.eulerAngles = Vector3.up *
Mathf.MoveTowardsAngle(transform.eulerAngles.y, targetRotation.eulerAngles.y,
rotationSpeed * Time.deltaTime);

        Vector3 input = new Vector3(Input.GetAxisRaw("Horizontal"), 0,
Input.GetAxisRaw("Vertical"));
        Vector3 motion = input;
        motion *= (Mathf.Abs(input.x) == 1 && Mathf.Abs(input.z) == 1) ? .7f : 1;
        motion *= (Input.GetButton("Run")) ? runSpeed : walkSpeed;
        motion += Vector3.up * -8;

        controller.Move(motion * Time.deltaTime);
    }

    void ControlWASD()
    {
        Vector3 input = new Vector3(Input.GetAxisRaw("Horizontal"), 0,
Input.GetAxisRaw("Vertical"));

        if (input != Vector3.zero)
        {
            targetRotation = Quaternion.LookRotation(input);
            transform.eulerAngles = Vector3.up *
Mathf.MoveTowardsAngle(transform.eulerAngles.y, targetRotation.eulerAngles.y,
rotationSpeed * Time.deltaTime);
        }

        Vector3 motion = input;
        motion *= (Mathf.Abs(input.x) == 1 && Mathf.Abs(input.z) == 1) ? .7f : 1;
        motion *= (Input.GetButton("Run")) ? runSpeed : walkSpeed;
        motion += Vector3.up * -8;

        controller.Move(motion * Time.deltaTime);
    }
```